Contents

Should You Use Oracle PDBs on AWS RDS? A CTO-Level Reality Check

Navigating the complexities of Oracle Multitenant in cloud-native environments

Executive Summary

As enterprises continue accelerating their move to the cloud, evaluating whether Oracle's Pluggable Database (PDB) architecture aligns with Amazon Web Services' managed database platform has become an important architectural consideration. This paper provides a detailed assessment of Oracle Multitenant support in AWS RDS, examining its technical capabilities, current limitations, and the practical factors enterprise architects and decision-makers must weigh before adoption.

This study takes a complete 360-degree view of deploying Oracle Pluggable Databases on AWS RDS and focuses on understanding their behaviour and boundaries in real-world use. It looks at how PDBs perform in areas such as administration, high availability, backup and

recovery, security, and upgrades. The intention is to give architects and decision makers a clear understanding of what works well and where challenges or restrictions can appear when using Oracle PDB in AWS RDS environments.

Key Takeaways:

- Oracle Multitenant is partially supported in AWS RDS with significant limitations
- Critical feature gaps exist around Data Guard, RAC, and advanced recovery options
- Strategic planning is essential for licensing compliance and operational efficiency
- Hybrid approaches may be necessary for enterprise-grade requirements

All the SQL examples and configuration scripts mentioned in this article can be found in the GitHub repository at https://github.com/asiandevs/OracleRDS.
This repository includes practical samples that you can use to test the concepts discussed here — such as RDS configuration, PDB and CDB management, and migration verification. It's meant to help readers easily try things out and understand how these features behave in real environments.

NOTE: **"Cloud vendor documentation is continuously updated, so always refer to the latest official documents to verify current features and changes."**

Understanding Oracle Multitenant Architecture in the Cloud Context

Oracle's Multitenant Architecture transformed traditional database management by allowing a single Container Database (CDB) to host multiple Pluggable Databases (PDBs). Each PDB functions as a logically isolated database while sharing memory, background processes, and storage within the same CDB. This approach enables higher resource efficiency, simplified administration, and faster provisioning compared to managing multiple standalone databases. The concept aligns strongly with cloud economics, where consolidation, elasticity, and cost optimization are central to achieving operational efficiency in large-scale environments.

The Value Proposition

Operational Efficiency:

- Streamlined patching and maintenance operations
- Reduced administrative overhead through centralized management
- Faster provisioning and deprovisioning of database environments

Resource Optimization:

- Consolidated memory and CPU utilization
- Shared background processes and infrastructure
- Improved hardware utilization ratios

Agility Benefits:

- Rapid PDB cloning for development and testing
- Seamless database migrations through PDB portability
- Simplified backup and recovery operations

AWS RDS Oracle Multitenant: Current Support Matrix

With the below compatibility matrix between Oracle Database versions and the non-CDB/CDB database type. Oracle 19c Supports both CDB and non-CDB configurations, providing flexibility for migration scenarios.

Oracle 21c and Later, Requires CDB architecture exclusively, aligning with Oracle's strategic direction toward multitenant-first deployments.

Feature	Oracle 19c	Oracle 21c	Notes
Container Database (CDB)	✓	✓	Full support
Non-CDB	✓	✗	Legacy support only
Initial PDB Count	1	1	Additional PDBs can be created post-deployment
Parameter Management	CDB-level	CDB-level	Parameters cascade to all PDBs
Bigfile Tablespaces	✓	✓	Default configuration

Critical Limitations and Architectural Constraints

1. MAX_PDBS Parameter Restrictions

Challenge: AWS RDS does not permit modification of the MAX_PDBS parameter, which may result in potential licensing compliance risks.

Impact: Under Oracle Standard Edition 2 (SE2), customers are entitled to use up to three PDBs per CDB. Deployments that exceed this limit without proper licensing can lead to:

- Unfavorable findings during Oracle license audits
- Financial penalties or compliance violations

Validation:

```
-- Monitor current PDB count
```

```
SELECT COUNT(*) as PDB_COUNT FROM V$PDBS WHERE NAME != 'PDB$SEED';

-- Verify MAX_PDBS setting
SELECT VALUE FROM V$PARAMETER WHERE NAME = 'max_pdbs';
```

2. Advanced Recovery Feature Gaps

Supported:

- Flashback Query
- Flashback Table
- Flashback Transaction

Not Supported:

- Flashback Database
- (Use **automated backups** or read replica promotion for PITR)

---

3. Disaster Recovery/Data Guard Support

Oracle Data Guard: Unavailable for CDB configurations, impacting disaster recovery strategies.

Alternative Approaches:

- Leverage AWS RDS automated backups for point-in-time recovery
- Implement cross-region read replicas for disaster recovery

Consider Oracle Data Pump for PDB-level recovery operations

---

4. High Availability / Real Application Clusters (RAC) Support

Real Application Clusters (RAC): Not supported in RDS, necessitating single-instance deployments.

Implications for Enterprise Workloads:

- Reduced availability during maintenance or patching events
- Limited scalability options for read-instensive workloads
- Potential performance bottlenecks for high-concurrency operations

Recommended Alternatives:

- AWS RDS Multi-AZ for automated failover and improved availability
- Amazon Aurora for cloud-native high availability and scalability

- Oracle on EC2 for customers requiring full RAC capabilities and configuration flexibility

---

Integration Challenges and Workarounds

AWS Database Migration Service (DMS) Considerations

Current Limitation: AWS Database Migration Service (DMS) does not automatically discover individual Pluggable Databases (PDBs) within a Container Database (CDB). Each PDB must be configured manually as a separate DMS endpoint.

Best Practice Implementation:

```
CDB Environment: PRODCDB
├── PDB1_SALES
├── PDB2_INVENTORY
└── PDB3_ANALYTICS

Required DMS Configuration:
- Endpoint 1: PRODCDB:1521/PDB1_SALES
- Endpoint 2: PRODCDB:1521/PDB2_INVENTORY
- Endpoint 3: PRODCDB:1521/PDB3_ANALYTICS
```

External Table Configuration

Requirement: Directory objects must be created using RDS-specific stored procedures rather than direct SQL create directory statements.

```
-- Create directory object
EXEC rdsadmin.rdsadmin_util.create_directory(
    p_directory_name => 'ETL_DATA_DIR'
);

-- Verify creation
SELECT * FROM dba_directories WHERE DIRECTORY_NAME = 'ETL_DATA_DIR';
```

SSL/TLS Encryption Implementation

Configuration Steps:

1. Add Oracle SSL option to the DB instance option group
2. Configure secondary listener port for encrypted connections
3. Update all application connection strings to use the SSL enabled endpoint

Security Considerations:

- AWS RDS supports both encrypted and clear-text connections simultaneously.

- Certificate management is fully handled by AWS, including rotation and renewal.
- Application-level SSL validation may require additional setup depending on the client.

Operational Best Practices

Memory Management Configuration

HugePages Implementation:

HugePages & Memory Settings

hugepages are enabled by default in Amazon RDS for Oracle to improve memory performance and reduce page table overhead. Automatic Memory Management (AMM) is not supported when Hugepages is active.

```
-- Verify HugePages configuration
SELECT value FROM v$parameter WHERE name = 'use_large_pages';

-- Check memory target settings
SELECT value FROM v$parameter WHERE name = 'memory_target';
SELECT value/1024 FROM v$parameter WHERE name = 'sga_target';
```

Required Settings:

- `memory_target` = 0
- `memory_max_target` = 0
- `sga_target` = Sized per instance class
- `use_large_pages` = ONLY

Custom Password Policy Implementation

Creating Enhanced Security Policies:

The default `ORA12C_STIG_VERIFY_FUNCTION` password verification function is enforced in Amazon RDS for Oracle and cannot be modified.

```
BEGIN
    rdsadmin.rdsadmin_password_verify.create_verify_function(
        p_verify_function_name => 'enterprise_verify_function',
        p_min_length => 14,
        p_min_uppercase => 2,
        p_min_lowercase => 2,
        p_min_digits => 2,
        p_min_special => 2,
        p_disallow_at_sign => true
```

```
    );
END;
/

ALTER PROFILE enterprise_profile LIMIT
    PASSWORD_VERIFY_FUNCTION enterprise_verify_function;
```

Backup and Recovery Strategy

Snapshot-Based Approach: Amazon RDS for Oracle relies on snapshot based mechanisms for backup and recovery management.

- Automated backups enable point-in-time recovery at the CDB level
- Manual snapshots for milestone-based recovery points
- Cross-region snapshot copies provide a for disaster recovery option for the region resilience.

PDB-Level Recovery:

```
-- Export PDB for granular backup
expdp system/password@pdb1 FULL=Y DIRECTORY=backup_dir
    DUMPFILE=pdb1_full_%U.dmp LOGFILE=pdb1_full.log;

-- Import PDB for recovery
impdp system/password@pdb1_new FULL=Y DIRECTORY=backup_dir
    DUMPFILE=pdb1_full_%U.dmp LOGFILE=pdb1_restore.log;
```

Migration Strategies and Considerations

Non-CDB to CDB Conversion: Check reediness before conversion and resolve any plugin violations.

For Oracle 19c Environments:

```
-- Check conversion readiness
SELECT * FROM PDB_PLUG_IN_VIOLATIONS WHERE STATUS != 'RESOLVED';

-- Perform conversion
EXEC DBMS_PDB.DESCRIBE(pdb_descr_file => '/tmp/ncdb.xml');
```

For Oracle 21c Migrations:

1. Non-CDB databases no longer supported, Convert non-CDB to CDB in Oracle 19c
2. Upgrade to Oracle 21c
3. Migrate to AWS RDS

Best Practices:

- Validate character set and compatibility before conversion
- Test in lower environments to confirm application behaviour.

- Use Data Pump for schema level migration where plug-in validations in place.

Licensing Optimization Strategies

Edition-Based Recommendations:

Standard Edition 2:

- Limited to a Maximum 3 PDBs per CDB
- Consider workload consolidation patterns
- Monitor CPU and memory utilization

Enterprise Edition:

- Leverage advanced features (Advanced Security, Partitioning and Diagnostic packs etc)
- Evaluate per-core vs. per-named-user licensing models
- Consider Oracle Cloud@Customer for hybrid deployments

Performance Optimization Techniques

Bigfile Tablespace Configuration

RDS uses **Oracle Managed Files (OMF)** and enables **bigfile tablespaces** by default (except TEMP tabelspaces).

Verification and Management:

```
-- Check tablespace configuration
SELECT tablespace_name, bigfile, allocation_type
FROM dba_tablespaces
ORDER BY tablespace_name;

-- Create bigfile tablespace
CREATE BIGFILE TABLESPACE app_data
    DATAFILE SIZE 10G AUTOEXTEND ON NEXT 1G MAXSIZE 100G;
```

Performance Benefits:

- Reduced file management overhead and simplified administration.
- Enhanced RMAN backup andrestore performance
- Support for datafiles up to 128TB size.

Database Link Configuration

Cross-PDB Connectivity: In Amazon RDS for oracle, database links enable cross-PDB connectivity with the same CDB or across different RDS instances.

```
-- Create database link between PDBs
CREATE DATABASE LINK pdb2_link
    CONNECT TO app_user IDENTIFIED BY password
    USING 'host:port/PDB2_SERVICE';

-- Test connectivity
SELECT * FROM dual@pdb2_link;
```

Other Features

Errors Granting Roles Like `UTL_MAIL`

Certain Oracle features (like Oracle Text or Spatial) must be enabled through the RDS option group before their packages can be used.
Packages such as UTL_MAIL.

Attempting to grant privileges or execute these packages without the required option result below error.

```
ORA-04042: procedure, function, package, or type is not allowed here
```

Granting SYS Object Privileges

In Amazon RDS for Oracle, direct grant statements owned by SYS-owned objects are not permitted.

Using standard SQL `GRANT` will result with the error: `ORA-04042`.

Instead, privileges must be granted using the `rdsadmin_util.grant_sys_object` procedure.

Example:

```
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object(
    p_obj_name  => 'V_$SESSION',
    p_grantee   => 'TEST',
    p_privilege => 'SELECT');
END;
/
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object(
    p_obj_name     => 'AUD$',
    p_grantee      => 'TEST',
    p_privilege    => 'SELECT',
```

```
    p_grant_option => true);
END;
/
```

No Archive Log Mode by Default

Unless **automated backups** are enabled, Oracle RDS runs in `NOARCHIVELOG` mode, which disables RMAN backups and point-in-time recovery.

```
SELECT LOG_MODE FROM V$DATABASE;
```

Enable backups to automatically switch to `ARCHIVELOG`.

Creation Dates May Be Misleading

Database creation timestamps in Amazon RDS for Oracle may not reflect the actual instance provisioning time. This is because of AWS provisions new databases using pre-build DBCA templates.

Timezone Considerations

- RDS instances defaults to UTC timezone.
- To change the timezone, modify the instances option group by adding or updating the Timezone option.
- Restart the instance for the change to take effect.

References: 🔗 AWS Timezone Guide

Additional Notes

Database Upgrades

Upgrade to Oracle Database 21c or later is supported only for CDB. If your instance is non-CDB then we must convert to CDB before performing upgrade.

OEM SQL Execution

Oracle Enterprise Manager cannot run SQL statements directly on on RDS instances. Because the RDS does not grant sys or os level access. Trough APIs basic monitoring possible.

Decision Framework for Enterprise Architects

Assessment Criteria

Technical Requirements:

- [ ] High availability requirements (RPO/RTO)
- [ ] Disaster recovery capabilities
- [ ] Performance and scalability needs
- [ ] Integration complexity

Operational Considerations:

- [ ] Administrative overhead
- [ ] Backup and recovery procedures
- [ ] Monitoring and alerting requirements
- [ ] Compliance and security needs

Financial Factors:

- [ ] Oracle licensing costs
- [ ] AWS infrastructure expenses
- [ ] Operational overhead
- [ ] Migration investment

Recommendation Matrix

Scenario	Recommended Approach	Rationale
Development/Testing	RDS with PDB	Cost-effective, managed service benefits
Production (Low HA)	RDS Multi-AZ with PDB	Adequate availability, reduced complexity
Production (High HA)	Oracle on EC2 with RAC	Full feature set, maximum availability
Hybrid Cloud	Oracle Cloud@Customer	Best of both worlds, consistent experience

---

Conclusion

Choosing whether to run PDB architecture on AWS RDS ultimately depends on what your organization values most. RDS delivers convenience with automated backups, patching, and maintenance, making it suitable for teams that prioritize managed services over deep database customization. However, these same advantages come with trade-offs. Many architectural and administrative operations are restricted, which can limit flexibility, recovery options, and performance tuning capabilities that enterprise workloads often require.

From a decision-making perspective, it's important to align platform choice with business priorities. If the goal is operational ease and reduced management effort, RDS provides a reliable and simplified path. But if your environment demands full control, scalability, and advanced database capabilities, a more flexible setup, such as Oracle on EC2, Exadata Cloud@Customer, or Oracle Cloud Infrastructure (OCI) may be a better strategic fit.

In short, PDBs can technically run on AWS RDS, but the decision should not be technical alone, it should weigh the balance between operational convenience and architectural freedom, ensuring the chosen platform truly supports long-term enterprise goals.

Key Recommendations:

1. **Evaluate your specific requirements** against RDS capabilities and limitations
2. **Develop a comprehensive migration strategy** that addresses licensing, performance, and operational needs
3. **Consider hybrid approaches** for enterprise-grade requirements that exceed RDS capabilities
4. **Invest in monitoring and observability** to ensure optimal performance and cost management
5. **Stay informed** about AWS and Oracle roadmap developments that may impact your architecture

---

About the Authors

Monowar Mukul is a senior cloud and solution architect with over 20 years of experience in enterprise technology transformation. He holds certifications as a Oracle Cloud Infrastructure Architect Professional, Microsoft Azure Solutions Architect Expert, AWS Solutions Architect – Professional and TOGAF Enterprise Architect, specialising in multi-cloud strategy, automation, and governance.

Nassyam Basha is an Oracle ACE Director and a well-known expert in database technologies. He has extensive experience in Oracle architecture, performance tuning, and large-scale migrations across banking, public sector, and healthcare industries. Over his career, he has delivered complex solutions involving Oracle Data Guard, GoldenGate, Exadata, and cloud migrations to OCI and multi-cloud environments.

He has authored multiple books and published numerous articles on Oracle technologies, and his work has been featured in Oracle Magazine. Nassyam is an active community leader, speaker, and mentor who contributes regularly to Oracle User Groups and industry events around the world. He is passionate about sharing practical knowledge, simplifying complex database concepts, and helping organizations build secure, high-performing, and resilient data platforms.

References:

- Amazon RDS for Oracle Documentation
- Oracle Multitenant Architecture Guide
- AWS Database Migration Service Best Practices
- Oracle Licensing in Cloud Environments